Forensic Science International: Digital Investigation 43 (2022) 301448

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS 2022 APAC - Proceedings of the Second Annual DFRWS APAC

Forensic investigation of Google Meet for memory and browser artifacts

Farkhund Iqbal [a], Zainab Khalid [b, *], Andrew Marrington [a], Babar Shah [a], Patrick C.K. Hung [c]

[a] College of Technological Innovation, Zayed University, Dubai, United Arab Emirates
[b] National University of Science and Technology (NUST), SEECS, Islamabad, Pakistan
[c] Ontario Tech University, Canada

ARTICLE INFO

Article history:

Keywords:
Browser forensics
Google Meet
Memory forensics
Video conferencing

ABSTRACT

Web applications have experienced a widespread adaptation owing to the agile Service Oriented Architecture (SOA) reflecting the ever-changing software needs of users. Google Meet is one of the top video conferencing applications, especially in the post-COVID19 era. Security and privacy concerns are therefore critical. This paper presents an extensive digital forensic analysis of Google Meet running on multiple browsers and software platforms including Google Chrome, Mozilla Firefox, and Microsoft Edge browsers in Windows 10 and Linux. Artifacts, traces of potential evidence, are extracted from different locations on a client's desktop, including the memory and browser. These include meeting records, communication records, email addresses, profile pictures, history, downloads, bookmarks, cache, cookies, etc. We explore how different Random Access Memory (RAM) sizes of client devices impact the persistence and format of extracted memory artifacts. A memory artifact extraction tool is developed to automate the extraction of artifacts identified via unstructured string analysis. Google Meet forensic artifacts are critical in that they are potential digital evidence in relevant criminal investigations. Additionally, they highlight that user data can be extracted despite implementing multiple privacy and security mechanisms.

1. Introduction

Video conferencing applications such as Zoom, Microsoft Teams, Cisco WebEx, and Google Meet have played a pivotal role in achieving the novel *work-from-home* norm on account of COVID19. According to a GetVoIP report, these applications had hundreds of millions of active users in 2020 (Stone,). This worldwide prevalence has attracted malicious agents' exploitation of security vulnerabilities. Attacks include meeting bombing, distributing malicious links in chats, stolen meeting links, and host privileges' transfer (Top videoconferencing attacks,).

Unlike other video conferencing (desktop client) applications, Google Meet is essentially a Web application. Web clients pose unique challenges for forensic analysts since they depend on agile SOA, which enables a dynamic Web infrastructure that deploys Web services and applications on a need basis, reflecting changing software requirements of users Akremi et al. (2020). This makes it harder to implement security controls that work efficiently on regular applications. Also, Web applications do not store client application folder(s) on disk, which may contain potential forensic artifacts. Browser data, *"residual data generated on devices, can be used as a proxy to data that is being stored in cloud environments,"* but it provides an incomplete understanding of artifacts pertinent to a Web client Cloyd et al. (2018); Case and Richard (2017). Browser cache stored on disk does not include meeting and communication records, which hold primary forensic relevance as digital evidence in investigations. Fortunately, these records can be carved from memory, which is fundamental for the extraction of volatile data that cannot be found on a disk/network or may exist in encrypted form.

Memory forensics typically focused on detecting rootkits and retrieving traces of malware (e.g., resident virus) from a system's volatile memory, has seen increasing research and development in techniques for extraction and analysis of userland application artifacts. Analysis of raw physical memory to extract application data

* Corresponding author.
E-mail address: zkhalid.msis18seecs@seecs.edu.pk (Z. Khalid).

https://doi.org/10.1016/j.fsidi.2022.301448

structures optimally requires the application to be open-source Schatz and Cohen (2017). Source code analysis makes it possible to construct accurate high-level data structures. However, most Windows and third-party applications like Zoom, MS Teams, Google Meet, and Cisco WebEx are not open-source. This leads to forensic analysts having to reverse engineer structures without proprietary code.

Additionally, the heterogeneity of applications requires this task to be performed individually for each application which may lead to a loss of time-cost relevance in most investigations. Consequently, *structured analysis* of userland applications may not be possible. Finally, string search to identify signatures of application data also poses multiple challenges like manual identification of signatures from huge memory dumps and incorporating changes in signatures with continuous updates of applications (Case and Richard (2017)). This is a daunting task to conduct and maintain, but it is a viable last resort. Once signatures are identified, operations can be automated for efficiency.

This research aims to perform an extensive memory and browser forensic analysis of the (closed-source) Google Meet Web client. Three major contributions of this study are presented as follows:

- An exhaustive (unstructured) memory forensic analysis of Google Meet to extract artifacts that contribute to a holistic meeting scenario and development of a memory artifact extraction tool to automate string signature-based artifact carving.
- Investigation of the impact of various client device RAM sizes on extracted memory artifacts.
- Analysis of browser artifacts of Google Meet extracted from Chrome, Firefox, and Edge.

2. Related work

Memory analysis for Web and desktop client artifacts. Barradas et al. (2019) extracted communication records of various Web clients and mobile applications (including Facebook, Messenger, Skype, Twitter, Outlook, Roundcube, Google Hangouts, WhatsApp, Telegram, Trillian, and Gmail) from physical memory using string analysis. According to reported results for Web clients, the latter 5 applications yielded no communication artifacts in Chrome. Similarly, 7 and 5 applications out of 11 yielded no results in Firefox and Edge. The experiments for Web clients were conducted in Virtual Machines (VMs) with RAMs of only 1 GB, which is inherently inadequate for real-world scenarios. The application data may likely be swapped out of memory onto disk (*pagefile.sys*) when it comes to devices with smaller RAMs of 1–4 GB, but this is unaddressed in the experiments and results of the paper. Today, client devices have RAMs ranging from 8 to 32 GB, which means they have significantly enhanced system load tolerances and application string data is bound to persist in memory and/or swap space. To this end, we sought to test the hypothesis that RAM sizes may have significant effects on the *persistence* and *format* of memory artifacts which cannot be overlooked while conducting experiments.

Fernández-Álvarez and Rodríguez (2022) employed the open-source code of the Telegram desktop client to extract artifacts prevalent in memory (user account information, communication records, contacts, etc.). They recreated the Unified Modeling Language (UML) diagram of Telegram using the source code, which helped identify how application objects were stored in memory. This gave an exact signature to search for, significantly eliminating error and chance of false positives in the extracted artifacts. The adopted methodology is an effective approach for investigations involving open-source applications. However, it cannot be applied to proprietary software.

Browser forensics. The client's browser is another source of forensic artifacts in cases involving Web clients. Cloyd et al. (2018) investigated residual data retained in a browser after a Facebook Web browsing session. Public browser modes of Chrome, Firefox, and Internet Explorer were reported to retain 46%, 61%, and 52% of activities performed in test sessions, respectively.

Marrington et al. (2012) tested the portable browser mode of Chrome (normal and private modes) to investigate whether privacy claims regarding portable browsers were legitimate. The authors extracted traces of Web browsing activity from the host's disk space and warned that users trying to obscure their online activity using portable browsers might not be using the most effective method.

Oh et al. (2011) emphasized that Web browser forensic analyses usually comprise log parsing only. The authors suggested that artifacts are likely spread out in different locations and an integrated analysis is necessary. Possible sources of evidence and different kinds of analyses such as timeline analysis, search history analysis, user activity analysis, and recovery of deleted information were discussed.

Forensics of video conferencing applications. Forensic analysis of video conferencing applications has been an active research topic recently. Mahr et al. (2021) performed Zoom's in-depth disk space forensic analysis. The authors explored client databases in the Zoom data directory to extract artifacts such as contacts, chats, email addresses, passwords, cache, and user/device configurations. Structured Query Language (SQL) queries used to extract relevant data from databases were also tabulated. In addition, preliminary memory and network analyses were presented.

Nicoletti and Bernaschi (2019) presented case studies illustrating the relevance of artifacts extracted from the Voice over Internet Protocol (VoIP) codec and protocols of Skype for Business. Nicoletti and Bernaschi (2021) also studied Microsoft Teams for disk space artifacts. The integration of Teams with Public Switched Telephone Network (PSTN) was also explored from a forensics perspective.

Bowling (2021) performed disk space forensic analysis of Microsoft Teams in Android, iOS, and Windows, extracting SQLite databases and analyzing the caching structure of Teams for artifacts.

Khalid et al. (2021) performed forensic analysis of Cisco WebEx in a Windows 10 Operating System (OS), investigating memory, disk space, and network artifacts. They extracted user account information, communication artifacts, passwords, etc.

Recent research work by Azhar et al. (2021) detailed forensic analysis of Microsoft Teams and Google Meet concerning disk space, memory, and network. Their analysis of Google Meet discussed Volatility's *pslist* and *netscan* outputs of memory dumps and *History* SQLite database on disk, therefore, we report no overlaps between our research work.

3. Google Meet Web client

Google Meet is an *on-the-go* video conferencing Web client in that the users can conduct quick meetings without downloading a desktop application. It offers three meeting scenarios: (1) *'Create a meeting for later,'* (2) *'Start an instant meeting,'* and (3) *'Schedule in Google Calendar'*, as shown in Fig. 1. In addition, users may join others' meetings by entering a code/nickname in the application User Interface (UI) or joining via an invitation email. No records of previous meetings and *in-call* messages are stored after the meeting, according to a note displayed atop the in-call message box in Google Meet: *'Messages can only be seen by people in the call and are deleted when the call ends.'* Only records of scheduled meetings

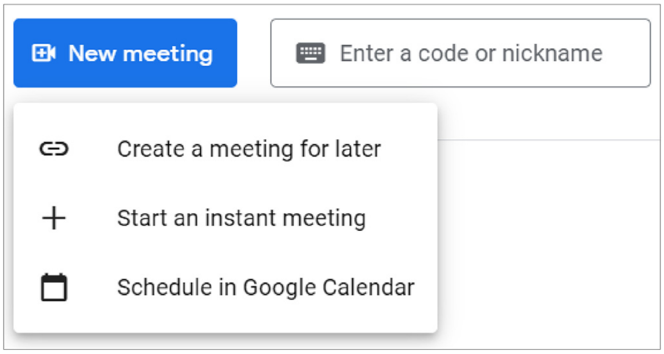

Fig. 1. Meeting options for Google Meet.

via Google Calendar are kept on the Calendar itself. Other features include screen sharing, captioning, and *whiteboarding*, which uses another Google application called *Jamboard*.

Google Meet offers attractive features for users with privacy concerns since it does not need to be downloaded and no meeting/in-call records are seemingly kept. In our forensic analysis of the Web client, we investigate whether communication records and other artifacts can still be extracted from memory and browser despite the privacy claims.

4. Experiments

Test environment. Three Windows 10 VMs with varying memory sizes of 4 GB, 8 GB, and 12 GB were created as testbeds to perform test activity, simulating the actions of a typical user of Google Meet. A Gmail account with its corresponding Google Meet Web client was set up in each VM. The test activity was performed on three different browsers: Chrome, Firefox, and Edge. Test OSs included Windows and Linux. To test on Linux, an additional VM of 8 GB RAM was created.

Test activities. The experiments conducted for forensic analysis of Google Meet Web client comprise test user activities which are categorized into *Test Activity Classes* (TAC):

- TAC_1: includes ‘Create a meeting for later’ and ‘Start an instant meeting’. In both scenarios, user activities include login, starting the meeting, exchange of 6 (3 sent and 3 received) in-call messages, screen share, closed captioning on, whiteboard activity using Jamboard, and *.pdf* and *.jpg* downloads of the activity.
- TAC_2: includes the ‘Schedule in Google Calendar’ scenario. User activities include login, scheduling the meeting using Google Calendar, starting the meeting, exchange of 6 (3 sent and 3 received) in-call messages, screen share, closed captioning on, whiteboard activity using Jamboard, and *.pdf* and *.jpg* downloads of the activity.
- TAC_3: includes joining a meeting set up by another user. User activities include login, joining the meeting, exchange of 6 (3 sent and 3 received) in-call messages, screen share, closed captioning on, whiteboard activity using Jamboard, and *.pdf* and *.jpg* downloads of the activity.

TACs were repeated in all created VMs. The VMs were restarted each time to perform successive TACs. Test activities were performed over a period of two months. Each TAC generated artifacts that are categorized into *Artifact Classes* (AC):

- AC_1 > Traces of Google Meet's usage
- AC_2 > Meeting records
- AC_3 > Communication records
- AC_4 > Document/image downloads
- AC_5 > Correspondence
- AC_6 > Closed captioning transcripts

Launching Google Meet in a browser tab creates a process named *chrome.exe* in memory. Memory pages allotted to the tab's process are released when the browser/tab is closed. We tested differences between extracted artifacts in both scenarios to explore artifacts' persistence: (1) when the meeting had ended but the browser/tab was still open and (2) after the browser/tab had been closed. These scenarios were tested for each TAC.

Memory was captured by suspending the VM and duplicating the *.vmem* file using AccessData Forensic Toolkit (FTK) Imager. These captures were taken for the two scenarios discussed above (for each TAC in every VM), i.e., the browser/tab open vs. closed scenarios.

Browser forensic analysis was performed specifically for Windows. Forensic images of the disk space were captured by imaging the *.vmdk* file of the VMs.

Effects of RAM sizes on persistence and format of extracted memory artifacts. To test our hypothesis that RAM sizes may have effects on extracted application artifacts (discussed in Section “Related work”), we conducted the same TACs on Windows VMs of varying sizes, i.e., 4 GB, 8 GB, and 12 GB.

Page smearing. While greater RAM sizes offer better device performance, they often lead to *page smearing*.[1] In our experiments, memory dumps were captured by suspending the VM and duplicating *.vmem* file. This prevented smearing from occurring and eliminated the issue in our analyses because the memory of the VM was frozen and the dump was captured instantly Case and Richard (2017).

In addition, acknowledging that not all investigations involve VMs but actual client devices, we performed 5 TAC experiments on a laptop host with 12 GB memory to observe the effects of smearing on extracted artifacts. We consider at least 1 of the 5 memory dumps taken from the host device to be smeared, since smearing generally occurs in systems with 8 GB RAM or more (or systems under high load) and almost all memory captures contain some degree of smear Case and Richard (2017).

Table 1 lists tools used for forensic analysis of Google Meet.

5. Memory forensics

Artifacts extracted via memory forensics contribute to a holistic picture of a Google Meet meeting from all TACs. Our analysis considered an artifact to be present in memory when it could be tied to Google Meet or other Personally Identifiable Information (PII) artifacts of the test account for attribution. If an artifact existed without any identifier, it was of no use in the investigation. Therefore, we considered such artifacts absent.

Running processes. Running processes (with execution timestamps) related to Google Meet were extracted from memory via Volatility and identified as *chrome.exe*, *firefox.exe*, and *msedge.exe* for each browser, respectively. Since every tab's process name is generic, and not indicative of whether the process belongs to Google Meet, a simple *yarascan* search was done using the ‘google meet’ search term to identify the PID of Google Meet's *chrome.exe*, *firefox.exe*, and *msedge.exe* processes. Running process results apply

[1] Page smearing is “*an inconsistency that occurs in memory captures when the acquired page tables reference physical pages whose contents changed during the acquisition process*” Case and Richard (2017).

Table 1
Tools used for forensic analysis of Google Meet.

Tool	Version	Usage
Windows 10 VM	10	Test environment
Linux	Debian 10.x	Test environment
Google Meet Web client	2021.5.1.1	Video conferencing Web client to test for forensic artifacts
AccessData FTK Imager	4.5.0.3	Create forensic images of memory and disk
Volatility	2.6	Memory dump analysis
PhotoRec	7.2	Carve *.jpeg* images
Autopsy	4.19.1	Disk image analysis
Strings	2.53	Manual string search
DB Browser for SQLite	3.12.1	Browse application databases on disk space
Chromecacheview	2.27	View Google Meet cache
Chromecookiesview	1.66	View Google Meet cookies
DCode	5.5.21194.40	Timestamp decoding

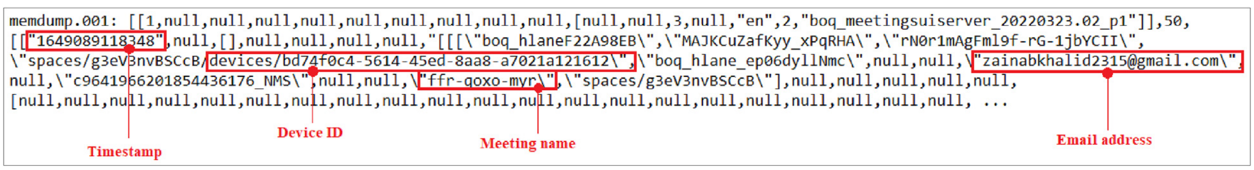

Fig. 2. Meeting record for TAC_1.

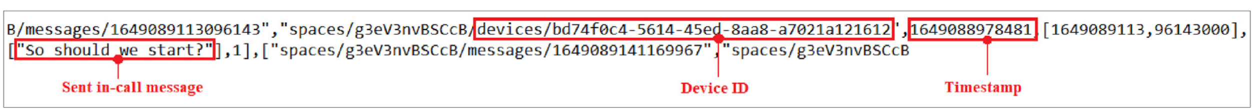

Fig. 3. Sent in-call message for 4 GB and 8 GB memory dumps of TAC_1.

to all TACs.

Profile photos and favicon images. Thumbnails of the user account avatars, interacted accounts, and other favicon images/logos related to Google Meet were carved from the memory dump using Photorec.[2] While favicons were a sound *trace of usage* of Google Meet, avatar thumbnails were not useful because they could not be associated with Google Meet or other PII. At best, the extracted profile avatars may provide a suspect list for the analyst to further investigate. The extracted images suggested that Google Meet stores profile images in unencrypted form (at least) in the memory before communicating them through the network (in contrast to the user account password, which was not found in memory in plaintext). Profile photos and favicon image results apply to all TACs.

Manual string analysis for extraction of meeting records. We performed a manual string analysis of the memory dumps using strings and grep tools. It is pertinent to note that results exhibited homogeneous signatures throughout our test browsers (Chrome, Firefox, Edge) and OSs (Windows, Linux). This confirmed that the format of high-level string data related to a Web client in memory depends on the application itself, independent of the browser/OS used, as reported by Barradas et al. (2019). Extracted manual artifacts presented in the following apply to all test browsers and OSs. Note that information related to an artifact (metadata) is represented in the *roster notation of sets* in this paper for a comprehensive yet compact presentation.

TAC_1. The string signatures of artifacts pertaining to ‘Create a meeting for later’ and ‘Start an instant meeting’ scenarios were similar; therefore, they are represented by TAC_1. A trace of usage found for these scenarios in memory was Google Meet link file(s): google meet.lnk.

AC_1 > Trace of Google Meet's usage = {.lnk file(s)}

Meeting records for TAC_1 were extracted with meeting names, email address of the test user, device ID, and timestamp of the meeting, as shown in Fig. 2. The email address and device ID serve as the PII of the test user.

AC_2 > Meeting records = {meeting name, email address, device ID, timestamp}

From 4 GB to 8 GB memory dumps of TAC_1, sent messages were recovered (extracted information included the in-call message, device ID, and timestamp), as shown in Fig. 3 However, received messages were not found in memory for these RAM sizes.

For 12 GB memory dumps of TAC_1, sent messages were recovered in a format different than that of 4 GB and 8 GB dumps. All messages sent by the test user were found collectively, in a single page line in memory along with the associated metadata (Fig. 4). This was in contrast to 4 GB and 8 GB dumps, where each of the sent in-call messages existed separately (with their metadata). In addition, received messages were also recovered in the 12 GB dumps, as shown in Fig. 5.

This difference in persistence and format of artifacts extracted from 4 GB, 8 GB, and 12 GB memory dumps confirmed our hypothesis that the amount of RAM available largely affects the artifacts extracted. Therefore, researchers and analysts must be wary of this in investigations.

AC_3 > Communication records = {sent/received in-call message, device ID, timestamp}

Document/image downloads via whiteboarding were identified for TAC_1 with the document/image name, the type of file (i.e., *.pdf* or

[2] https://www.cgsecurity.org/wiki/PhotoRec.

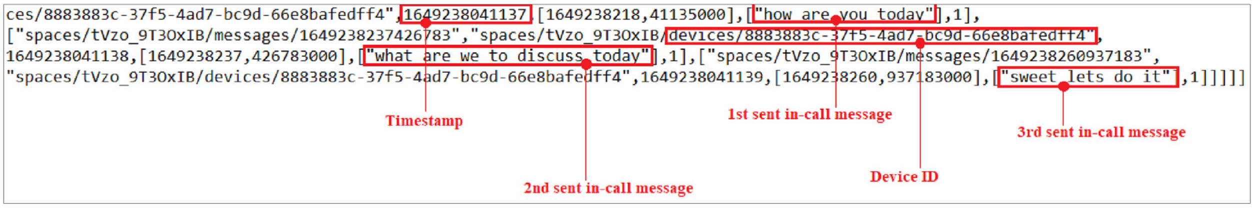

Fig. 4. Sent in-call message for 12 GB memory dump of TAC_1.

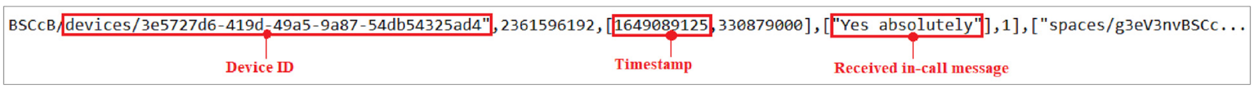

Fig. 5. Received in-call message for 12 GB memory dump of TAC_1.

.png), size of the file, email address of the test user, Jamboard link used to perform the whiteboard activity, and directory path of the stored file.

AC_4 > Document/image downloads = {document name, type, size, email address, directory path, Jamboard link}

The email addresses of other accounts the test user interacted with were extracted in a format that proved that the corresponding account was part of a Google Meet meeting. However, it could not be tied to a specific meeting.

AC_5 > Correspondence = {email address}

Closed captioning transcripts were found in memory but without any specific string signature/format.

AC_6 > Closed captioning transcripts = {}

TAC_2. Artifacts extracted for TAC_2 were majorly similar to TAC_1, as expected. However, the difference existed in AC_2, i.e., the extracted meeting records where the meeting title of the scheduled meeting, as set in Google Calendar, was also extracted.

AC_2 > Meeting records = {meeting title set in Google Calendar, meeting name, email address, device ID, timestamp}

In addition, *received* in-call communication records for 12 GB memory dumps of TAC_2 were also extracted in a collective/chained format as discussed for sent in-call messages of 12 GB dumps of TAC_1.

TAC_3. The artifacts extracted for TAC_3 (test user joining a meeting set-up by another user) were less detailed in the case of certain ACs, i.e., AC_3 and AC_4, compared to previous TACs. The remaining ACs of TAC_3 were similar to TAC_1 and TAC_2.

Communication records extracted for all RAM sizes were in the format: l,1651046186694,null,["<message>"],1]]. This only divulged the sent/received in-call message along with the timestamp. Sent in-call messages were also recovered in the form of <div> HTML tag clippings: up today">what is up today</div></div></div>. However, lack of signatures for extraction rendered this format useless.

AC_3 > Communication records = {sent/received in-call message, timestamp}

Information extracted related to AC_4 (document/image downloads) was also comparatively limited, as shown below.

AC_4 > Document/image downloads = {document name, type, directory path, Jamboard link}

Browser/tab open vs. closed. For memory dumps captured for all TACs, we observed whether extractable artifacts persisted in memory after the browser had been closed. Our analyses revealed that all ACs were extractable with no difference except AC_3 (communication records) and AC_5 (correspondence). Some sent and received in-call messages were absent and the interacted account's email address was absent after the browser was closed. These results apply to all TACs.

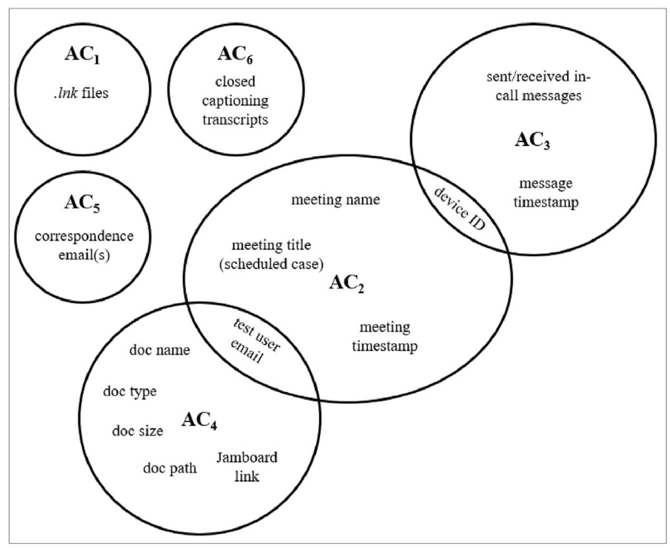

Fig. 6. Google Meet forensic artifacts.

Fig. 6 illustrates a holistic diagram of all forensic artifacts pertinent to a Google Meet meeting. The diagram represents all TACs in a general manner. While AC_1 provides a trace of usage for Google Meet, AC_5 and AC_6 cannot be tied to a meeting (directly) via PII in any case.

Page smearing. Artifacts extracted from memory dumps of the laptop host exhibited similar results as observed in *.vmem* memory dumps discussed in the previous section. Page smearing did not majorly affect the extraction of memory artifacts in this case. This was expected because manual analyses of string-based userland applications' artifacts are based on a signature-matching algorithm, extracting artifacts if matched with a pre-defined signature. Once signatures are identified in the research phase, the related artifacts can generally be extracted from wherever they exist in the memory even when the process memory allocated to the subject process is smeared.

If page smearing causes inconsistencies mid-page line, artifacts may be extracted in clipped form. But this assumption was not identified in any of the 5 memory dumps acquired from the laptop host as the artifacts extracted were complete and consistent with the ones extracted from *.vmem* memory dumps.

Memory artifact extraction tool. Our python-based proof of

```
{
    "key": "b'\\x00\\x02\\x01\\x01\\x01\\x13\\x00c\\x00l\\x00i\\x00e\\x00n\\x00t\\x00R\\x00e\\x00g\\x00i\\x00s\\x00t\\x00r\\x00a\\x00t\\x00i\\x00o\\x00n\\x00s'",
    "origin_file": "C:\\Users\\HP\\AppData\\Local\\Google\\Chrome\\User Data\\Default\\IndexedDB\\https_meet.google.com_0.indexeddb.leveldb\\000003.log",
    "seq": 587,
    "state": "KeyState.Live",
    "store": "meet_store",
    "value": {
        "data": "[[[\"8060b862-3140-4252-a017-d4c4cc11bb04\",0,false,
                 [\"boq_hlane_I92507BX684\",
                 \"spaces/dnZOx4ufg04B/devices/e87e30bc-0076-4e42-93d4-dff5186880b7\",
                 null,\"tie-umyp-nio\",\"Test Meeting #1\",false,true,false]]]]",
        "key": "clientRegistrations"
    }
```


Fig. 7. Meet_store object with scheduled meeting link and title extracted via Google Meet IndexedDB-LevelDB.

concept memory artifact extraction tool[3] automates the extraction of manual string artifacts pertaining to Google Meet, employing the signatures identified in memory forensic analysis.

6. Browser forensics

Disk images captured after test user activity with Google Meet were analyzed using Autopsy to extract forensic artifacts. Our target artifacts in browser forensics included traces of the application's usage, history, downloads, bookmarks, cache, cookies, and relevant Uniform Resource Locators (URLs). Other artifacts such as associated profile pictures, email addresses, meeting links, and in-call messages were also extracted and documented from the disk image.

Traces of usage. Traces of Google Meet's usage from the browser were detected using several artifacts. In the Web Applications folder (*AppData\Local\Google\Chrome\User Data\Default\Web Applications\[.*]*), a folder containing the Google Meet icon and its md5 hash was extracted. No other information was stored in the folder; however, it is an indicator of the Web application's usage on the client device. Subjectively, Google Meet was found in the icons of the recently closed sites folder (*AppData\Local\Google\Chrome\User Data\Default\JumpListIconsRecentClosed*) and in the most visited sites folder (*AppData\Local\Google\Ch-rome\User Data\Default\JumpListIconsMostVisited*) depending on the frequency of usage. SQLite database Top Sites in *AppData\Local\Google\Chrome\User Data\Default* also listed Google Meet as one of the top sites the user engaged with; also subjective.

IndexedDB-levelDB folder. The IndexedDB folder at *AppData\Local\Google\Chrome\User Data\Default\IndexedDB* stores levelDBs of Web applications used by the user. LevelDB is a novel key-value structured database which stores session data related to a Web application (Caithness,). Once Google Meet was used, a folder, *https_meet.google.com0_.indexeddb.leveldb*, was created in the IndexedDB directory. This is a solid trace of usage, unlike prior indicators. After converting the IndexedDB-levelDB into a readable format (*.json*[4]), an analysis of the structure's storage format revealed that in the Google Meet levelDB, two *object stores*, namely IndexedStorage and meet_store were classified. IndexedStorage was found to be of no forensic relevance since it mainly logged *.woff2* font packages for the application. On the other hand, the meet_store revealed meeting IDs of all the previously held and joined meetings by the user, along with the GUID of the user. In case the meeting was a scheduled meeting via Google Calendar, the meeting title was consequently stored in the database as well, as shown in Fig. 7. It is pertinent to note that the object store did not store timestamps along with meeting IDs therefore the extent of forensically relevant clues the Google Meet levelDB may divulge is ‘whether a suspect used Google Meet or not?’ and ‘was (s)he part of a certain meeting or not?‘.

PII. The email address associated with the Google Meet account was extracted from the Login Data and Web Data SQLite databases. The avatar associated with Google Meet was extracted from *AppData\Local\Google\Chrome\User Data\Default\Accounts\Avatar Images*. Note that this is essentially the profile picture of the Google/Gmail account associated with Google Meet. If more than one avatars exist, it indicates usage of more than one Google account in which case attribution becomes trickier. This problem can be solved using cache entries of avatars discussed later.

Communication records. In Google Meet, an attractive feature is that the history of meetings and exchanged in-call messages is not recorded anywhere on the Web application (apart from meetings that are scheduled using Google Calendar, in which case it is possible to track down the history (meeting names) of scheduled meetings via Google Calendar). However, we identified logs in the Google Chrome data directory that stored information related to meetings conducted, including in-call messages exchanged. The Sessions folder (*AppData\Local\Google\Chrome\User Data\Default\Sessions*) stored logs of Web applications, tabs, and sessions. From the [App_#] logs, the Google Meet meeting links and in-call messages (following the <chatTextInput> and <textarea> tags) were extracted. The in-call messages extracted were scattered and fragmented, rendering the extraction a highly manual task, but in cases where messages play a pivotal role and capturing memory is not possible, parsing the logs is an option.

Browser artifacts. Google Meet bookmark saved in the Chrome browser was extracted from (*AppData\Local\Google\Chrome\User Data\Default\Bookmarks*) with a timestamp, ID, and name of the bookmark.

The history of Google Meet meetings extracted from the History SQLite database stored in *AppData\Local\Google\Chrome\User Data\Default\History* contained not only the browsing history (with timestamps, visit counts, and durations of visits) but also the keyword search terms entered in the browser and downloads (with names of files downloaded, sizes, start and end times of download, referrer URLs from which they were downloaded, and paths to folders they were saved in). Details regarding downloaded files were also found in the Download Metadata file in *AppData\Local\Google\Chrome\User Data\Default\Download Metadata*.

In case a suspect deletes the browsing history directly from the browser, it effectively clears all tables in the History database except for the *downloads*, *downloads_url_chains* and *url* tables.

The cookies related to Google Meet were extracted from *AppData\Local\Google\Chrome\User Data\Default\Network\Cookies* and *AppData\Local\Google\Chrome\User Data\Default\Safe Browsing Network\Safe Browsing Cookies* with names, host keys, values, creation times, expiration times, and last accessed times.

The cache folder (*AppData\Local\Google\Chrome\User Data\Default\Cache*) stored multiple Google Meet artifacts of forensic relevance. Profile pictures of the user and accounts the user interacted with were found in formats: *[.*]-[.*]-[.*].jfif*, *photo.jpg.jfif*, and *.png*. The extracted profile pictures were associated with the

[3] https://github.com/farkhund/googlemeet.
[4] https://github.com/lxndrblz/forensicsim.

Table 2
Directory paths for pertinent Google Chrome browser artifacts.

Artifacts	Directory paths
History	AppData\Local\Google\Chrome\User Data\Default\History
Bookmarks	AppData\Local\Google\Chrome\User Data\Default\Bookmarks
Cache	AppData\Local\Google\Chrome\User Data\Default\Cache
Cookies	AppData\Local\Google\Chrome\User Data\Default\Network\Cookies, AppData\Local\Google\Chrome\User Data\Default\Safe Browsing Network\Safe Browsing Cookies
IndexedDB-levelDB	AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_meet.google.com_0.indexeddb.leveldb
Downloads	AppData\Local\Google\Chrome\User Data\Default\Download Metadata
Profile picture	AppData\Local\Google\Chrome\User Data\Default\Accounts\Avatar Images
Email address	AppData\Local\Google\Chrome\User Data\Default\Login Data
Browsing sessions	AppData\Local\Google\Chrome\User Data\Default\Sessions
Traces of usage	AppData\Local\Google\Chrome\User Data\Default\Web Applications\[.*], AppData\Local\Google\Chrome\User Data\Default\JumpListIconsRecentClosed, AppData\Local\Google\Chrome\User Data\Default\JumpListIconsMostVisited, AppData\Local\Google\Chrome\User Data\Default\Top Sites

Table 3
Directory paths for pertinent Mozilla Firefox browser artifacts.

Artifacts	Directory paths
History	AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\places, AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\formhistory
Bookmarks	AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\places
Cache	AppData\Local\Mozilla\Firefox\Profiles\[#].default-release\cache2, AppData\Local\Mozilla\Firefox\Profiles\[#].default-release\jumpListCache
Cookies	AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\cookies
Downloads	AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\storage\default\https+++jamboard.google.com
Profile picture	AppData\Local\Mozilla\Firefox\Profiles\[#].default-release\jumpListCache
Traces of usage	AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\favicons, AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\storage\default, AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\AlternateServices, AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\enumerate_devices, AppData\Roaming\Mozilla\Firefox\Profiles\[#].default-release\SiteSecurityServiceState

Table 4
Directory paths for pertinent Microsoft Edge browser artifacts.

Artifacts	Directory paths
History	AppData\Local\Microsoft\Edge\User Data\Default\History
Bookmarks	AppData\Local\Microsoft\Edge\User Data\Default\Bookmarks
Cache	AppData\Local\Microsoft\Edge\User Data\Default\Cache\Cache_Data
Cookies	AppData\Local\Microsoft\Edge\User Data\Default\Cookies, AppData\Local\Microsoft\Edge\User Data\Default\Safe Browsing Cookies
IndexedDB-levelDB	AppData\Local\Microsoft\Edge\User Data\Default\IndexedDB\https_meet.google.com_0.indexeddb.leveldb, AppData\Local\Microsoft\Edge\User Data\Default\Local Storage\leveldb
Downloads	AppData\Local\Microsoft\Edge\User Data\Default\History
Profile picture	AppData\Local\Microsoft\Edge\User Data\Default\Cache\Cache_Data
Email address	AppData\Local\Microsoft\Edge\User Data\Default\Web Data
Browsing sessions	AppData\Local\Microsoft\Edge\User Data\Default\Sessions
Traces of usage	AppData\Local\Microsoft\Edge\User Data\Default\JumpListIconsRecentClosed, AppData\Local\Microsoft\Edge\User Data\Default\Service Worker\Database, AppData\Local\Microsoft\Edge\User Data\Default\Favicons, AppData\Local\Microsoft\Edge\User Data\Default\Network Action Predictor, AppData\Local\Microsoft\Edge\User Data\Default\Shortcuts, AppData\Local\Microsoft\Edge\User Data\Default\Top Sites

corresponding Google Meet URLs making them an effective attribution artifact. Other icons in the cache included Google Meet logos. ‘join call’ and ‘leave call’ audio tunes were cached. Other links pertaining to Jamboard sessions and Google Calendar were also found in the cache, given either application was used in correspondence to Google Meet. A meeting scheduled via Google Calendar specifically stored all the information in the cache regarding the meeting like meeting/conference ID, name and description of the meeting, location, creator and attendees' email addresses, start and end timestamps of meeting, time zones, call UID, and HTML link. The cache also included application UIs with operator parameters, meeting settings, and searches made using the browser. Some interesting cache entries found were various location predictions of the user during meetings that were conducted using maps by Google. The server IP addresses, server names, last accessed timestamps, and expiration timestamps of cached entries were also recovered. Note that if a suspect manually deletes the cache from the data directory, it effectively deletes all Google Meet cache artifacts.

Similar browser forensic analyses of Firefox and Edge were performed. In the case of Firefox, the major source of artifacts was the places SQLite database, which revealed the history (and metadata) of meetings conducted using Google Meet. It is pertinent to note that Chrome and Edge are built on Chromium's same underlying technology. On the other hand, Firefox operates on the

Quantum engine built specifically for the browser. This results in Chrome and Edge having similar data directory structures.

Tables 2–4 detail the directory paths of every artifact found in each browser's data directories. Evidently, Firefox does not have some evidence sources such as an IndexedDB-levelDB folder owing to its different browser engine and data directory structure.

7. Case study

A forensic analyst investigating a case of an insider attack targeting confidential company data was able to acquire memory and disk images of a suspect employee, *Eve's* laptop PC. In order to prove Eve's communication link with *Bob*, who was identified earlier to be in contact with the insider, forensic images from Eve's device were analyzed.

Google Meet, as one of the running applications on the device, was further explored in the Chrome data directory. A record of Eve's meetings (with IDs) conducted via Google Meet was recovered from the application's levelDB. The meeting IDs were further explored in the memory dumps. Timestamps of the meetings along with sent and received in-call messages (also with timestamps) were carved. Emails of accounts in correspondence with Eve were also recovered. These artifacts correspond to AC_2, AC_3, and AC_5:

AC_2 > Meeting records = {meeting name, email address, device ID, timestamp}

AC_3 > Communication records = {sent/received in-call message, device ID, timestamp}

AC_5 > Correspondence = {email address}

While Bob's email address was extracted in AC_5 (which proved Eve was in contact with Bob through Google Meet), the specific in-call messages received by Eve were only identifiable via the device ID as PII. In order to prove the received messages were sent by Bob, his email address (from AC_5) and the device ID (from AC_3) needed to be linked. Fortunately, this information was cross-checked from memory dumps taken from Bob's device (AC_2 from Bob's device tied both his email address and device ID together).

8. Conclusion and future work

Web applications are an efficient solution to the dynamic software needs of today's users. However, this dynamic nature presents challenges in the implementation of security controls, thereby increasing the attack surface. Our research aimed to perform a detailed forensic analysis of Google Meet to extract memory and browser artifacts that may serve as evidence in a court of Law.

We conducted an in-depth memory forensic analysis of Google Meet employing manual string analysis to extract traces of usage, detailed meeting records, communication records, information related to whiteboard activity downloads, and correspondence emails. We also explored the effects of various client device RAM sizes and page smearing on the extracted memory artifacts. In addition, we developed a memory artifact extraction tool to automate the extraction of the string signature-based artifacts.

This study also presented an exhaustive browser forensic analysis of Google Meet on Google Chrome, Mozilla Firefox, and Microsoft Edge extracting traces of usage, history, downloads, bookmarks, cache, cookies, profile picture, email addresses, meeting information, and in-call message logs related to the Web application.

This work can be further extended in multiple directions. Other OSs such as macOS, Android, and iOS may be tested for Google Meet forensic artifacts. Other Web clients and video conferencing applications can be put to the test of forensic analysis to investigate information they give away, being a pivotal element as evidence in criminal investigations.

Acknowledgement

This research is supported by Research Incentive Funds (R21111), and Provost Research Fellowship Award (R20093), Zayed University, United Arab Emirates.

References

Akremi, A., Sallay, H., Rouached, M., Bouaziz, R., 2020. Applying digital forensics to service oriented architecture. Int. J. Web Serv. Res. 17, 17–42. https://doi.org/10.4018/IJWSR.2020010102.

Azhar, H., Timms, J., Tilley, B., 2021. Forensic investigations of google meet and microsoft teams – two popular conferencing tools in the pandemic. In: 12th EAI International Conference on Digital Forensics and Cyber Crime.

Barradas, D., Brito, T., Duarte, D., Santos, N., Rodrigues, L., 2019. Forensic analysis of communication records of messaging applications from physical memory. Comput. Secur. 86, 484–497. https://doi.org/10.1016/j.cose.2018.08.013. URL: https://www.sciencedirect.com/science/article/pii/S0167404818311313.

Bowling, H.R., 2021. A Forensic Analysis of Microsoft Teams. https://doi.org/10.25394/PGS.15091329.v1. URL: https://hammer.purdue.edu/articles/thesis/A_Forensic_Analysis_of_Microsoft_Teams/15091329.

Caithness, A.. Hang on! that's not sqlite! chrome, electron and leveldb. URL: https://www.cclsolutionsgroup.com/post/hang-on-thats-not-sqlite-chrome-electron-and-leveldb. (Accessed 23 September 2021).

Case, A., Richard, G.G., 2017. Memory forensics: the path forward. Digit. Invest. 20, 23–33. https://doi.org/10.1016/j.diin.2016.12.004. special Issue on Volatile Memory Analysis. URL: https://www.sciencedirect.com/science/article/pii/S1742287616301529.

Cloyd, T., Osborn, T., Ellingboe, B., Glisson, W.B., Choo, K.K.R., 2018. Browser analysis of residual facebook data. In: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering. TrustCom/BigDataSE), pp. 1440–1445. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00200.

Fernández-Álvarez, P., Rodríguez, R.J., 2022. Extraction and analysis of retrievable memory artifacts from windows telegram desktop application. Forensic Sci. Int.: Digit. Invest. 40, 301342. https://doi.org/10.1016/j.fsidi.2022.301342. URL: https://www.sciencedirect.com/science/article/pii/S2666281722000117. selected Papers of the Ninth Annual DFRWS Europe Conference).

Khalid, Z., Iqbal, F., Kamoun, F., Hussain, M., Khan, L.A., 2021. Forensic analysis of the cisco webex application. In: 2021 5th Cyber Security in Networking Conference. CSNet, pp. 90–97. https://doi.org/10.1109/CSNet52717.2021.9614647.

Mahr, A., Cichon, M., Mateo, S., Grajeda, C., Baggili, I., 2021. Zooming into the pandemic! a forensic analysis of the zoom application. Forensic Sci. Int.: Digit. Invest. 36, 301107. https://doi.org/10.1016/j.fsidi.2021.301107. URL: https://www.sciencedirect.com/science/article/pii/S2666281721000019.

Marrington, A., Baggili, I.M., Ismail, T.A., Kaf, A.A., 2012. Portable web browser forensics: a forensic examination of the privacy benefits of portable web browsers. In: 2012 International Conference on Computer Systems and Industrial Informatics, pp. 1–6. https://doi.org/10.1109/ICCSII.2012.6454516.

Nicoletti, M., Bernaschi, M., 2019. Forensic analysis of microsoft skype for business. Digit. Invest. 29, 159–179. https://doi.org/10.1016/j.diin.2019.03.012. URL: https://www.sciencedirect.com/science/article/pii/S1742287618303967.

Nicoletti, M., Bernaschi, M., 2021. Forensics for microsoft teams. URL: https://arxiv.org/abs/2109.06097.

Oh, J., Lee, S., Lee, S., 2011. Advanced evidence collection and analysis of web browser activity. Digit. Invest. 8, S62–S70. https://doi.org/10.1016/j.diin.2011.05.008. URL: https://www.sciencedirect.com/science/article/pii/S1742287611000326 (the Proceedings of the Eleventh Annual DFRWS Conference).

Schatz, B., Cohen, M., 2017. Advances in volatile memory forensics. Digit. Invest. 20, 1. https://doi.org/10.1016/j.diin.2017.02.008. special Issue on Volatile Memory Analysis. URL: https://www.sciencedirect.com/science/article/pii/S1742287617300932.

Stone, K.. The state of video conferencing in 2020 [50 statistics] – getvoip. URL: https://getvoip.com/blog/2020/07/07/video-conferencing-stats/. (Accessed 7 July 2021).

Top Videoconferencing Attacks and Security Best Practices. URL:https://www.cisecurity.org/blog/top-videoconferencing-attacks-and-security-best-practices/. accessed: 2021, 10, 26.